NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

UNIVERSITY COURSE TIMETABLING WITH PROBABILITY COLLECTIVES

by

Brian Autry

March 2008

Thesis Advisor:	Kevin Squire
Second Reader:	Craig Martell

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2008	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE University Course Timetabling with Probability Collectives	**5. FUNDING NUMBERS**
6. AUTHOR Brian Autry	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200words)

The Naval Postgraduate School currently uses a time consuming manual process to generate course schedules for students and professors. Each quarter, the process of timetabling approximately 2000 students into nearly 500 courses takes up to 8 weeks. This thesis introduces an automated timetabling algorithm using Probability Collectives (PC) theory. PC Theory is an agent based approach that utilizes Collective Intelligence (COIN) to solve optimization problems by using a collection of agents attempting to achieve a single goal. The algorithm was tested on a set of data provided by the organizers of the 2007 International Timetabling Competition. The algorithm provided valid timetables for every problem instance and successfully scheduled between 70% and 91.6% of all student course requests.

14. SUBJECT TERMS Timetabling, Collective Intelligence, Probability Collective, Scheduling, Optimization	**15. NUMBER OF PAGES** 55
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

UNIVERSITY COURSE TIMETABLING WITH PROBABILITY COLLECTIVES

Brian M. Autry
Lieutenant, United States Navy
B.S. Computer Science, Texas A&M University, 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2008

Author: Brian M. Autry

Approved by: Kevin M. Squire
Thesis Advisor

Craig Martell
Second Reader

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Naval Postgraduate School currently uses a time consuming manual process to generate course schedules for students and professors. Each quarter, the process of timetabling approximately 2000 students into nearly 500 courses takes up to 8 weeks. This thesis introduces an automated timetabling algorithm using Probability Collectives (PC) theory. PC Theory is an agent based approach that utilizes Collective Intelligence (COIN) to solve optimization problems by using a collection of agents attempting to achieve a single goal. The algorithm was tested on a set of data provided by the organizers of the 2007 International Timetabling Competition. The algorithm provided valid timetables for every problem instance and successfully scheduled between 70% and 91.6% of all student course requests.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABRREVIATIONS

COIN	Collective Intelligence
NPS	Naval Postgraduate School
PATAT	Practice and Theory of Automated Timetabling
PC	Probability Collective
WATT	Working Group on Automated Timetabling

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my thesis advisor Kevin Squire for providing the idea and motivation for this thesis. He demonstrated patience and a successful teaching style while assisting me in learning the underlying theory.

I would also like to thank the organizers of the 2007 International Timetabling Competition. The resources provided by this competition proved invaluable during my research and served as a basis on which to test my algorithm.

Finally I would like to thank my mother, Carol Autry, and my children, Connor Autry and Samuel Autry, for their unwavering love and support.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

This research was prompted by the current course scheduling problem at the Naval Postgraduate School (NPS). Currently there are approximately 2000 students enrolled at NPS and nearly 500 courses offered each quarter. Scheduling is done manually by two people and the process takes up to 8 weeks to complete the schedule for one quarter. Clearly a better system is needed.

Most students at NPS are members of the military and are given a fixed number of months to complete their degree. A demand based scheduling approach is used to ensure that each student's needs are met. In the past, some of the complexity of developing schedules was reduced by assigning students a template of courses for each quarter. This allowed the scheduler to assign students in large blocks but reduced the flexibility each student had in customizing their education. The current system allows students to tailor their schedule placing additional burden on the scheduler.

NPS has tested several commercial scheduling applications but has yet to find an acceptable solution that meets all of its needs. Several NPS students have also conducted research relating to the problem. One student thesis involved the use of integer linear programming to create course schedules [1]. The research was moderately successful on small subsets of data but required significant run times and never found an optimal solution.

Another paper examined potential solutions to the problem by converting the scheduling system from a demand based system to a supply based system [2]. Although the proposed solution would allow for faster schedule generation, the flexibility desired by NPS would not be achieved and the requirement that every student would get the classes that they need when they need them could not be satisfied.

The same demand based scheduling concepts used in the timetabling of university courses are used in a broad array of applications. Timetabling algorithms have been used

to schedule personnel in laboratories [3] and for scheduling clinical rounds in hospitals [4]. The theory has also been applied to the scheduling of sports teams in tournament brackets [5] and in the delivery of goods in time-critical applications [6]. One of the goals of this thesis is to provide a generic algorithm that can be modified to help solve a wide range of problems.

B. INTERNATIONAL TIMETABLING COMPETITION

The 2nd International Timetabling Competition sponsored by Practice and Theory of Automated Timetabling (PATAT) and the Working Group on Automated Timetabling (WATT) was held in 2007 and consisted of three tracks: examination timetabling, post enrollment based course timetabling, and curriculum based course timetabling [7]. The post enrollment and curriculum based course timetabling tracks are both subsets of the course timetabling problem. The post enrollment course timetabling track closely models the system used at NPS and was used as a simplified example to test and evaluate the effectiveness of PC theory in solving this type of problem.

The post enrollment based timetabling problem consists of scheduling a set of n events (courses) into 45 timeslots (5 days, 9 hours per day). A set of r rooms exists each with a set of f room-features. A set s of students who attend a varying combination of events is provided. Each of the n events has a set of available timeslots. A set of requirements is also provided that determine which events should occur before other events. The goal is to schedule each event n into one of the r rooms and one of the timeslots while satisfying the following hard constraints:

- No student should be scheduled for two events at the same time.
- The room assigned to an event should be large enough for all of the students assigned to that event and should satisfy all of the room-features required by that event.
- Only one event is scheduled into each room in any timeslot.
- Events should only be scheduled in available timeslots.

- Events should be scheduled in the proper order as specified by any precedence requirements.

In addition to the hard constraints, three soft constraints were also specified:

- Students should not be scheduled for an event occurring at the end of the day.
- Students should not have to attend three or more events in a row.
- Students should not be required to attend only one event on a particular day.

No hard constraints can be violated or the solution is rejected. Since for some problem instances it may not be possible to schedule each event and maintain all of the hard constraints, certain events may need to be left out to ensure all hard constraints are satisfied. A timetable which does not have any hard constraint violations but leaves out some events is considered valid. A feasible timetable is one in which there are no occurrences of any hard constraint violations and all events are scheduled.

Solutions submitted for the competition are evaluated by first ensuring that they are valid. Next, a Distance to Feasibility is calculated by summing up the number of students in each unscheduled event. Finally a Soft Cost is calculated by summing the total number of occurrences of soft constraint violations listed above. The solution with the lowest Distance to Feasibility is winner. If two valid solutions have the same Distance to Feasibility, the solution with the lowest Soft Cost is judged the winner.

This problem description greatly simplifies the real world timetabling problem that exists at NPS. Some of the constraints and considerations not addresses by this problem formulation include:

- Multiple sections of the same course. This actually adds flexibility to the timetabling process by allowing the student's course requirement to be fulfilled in different timeslots.
- Multiple professors for the same course. This problem description considers professors linked to individual courses. At NPS, some courses are taught by multiple professors, particularly when considering labs associated with a course.

- Courses that are taught via online methods. These courses still have students assigned, but do not require a room.
- Room availability. Often rooms are prescheduled for events that are not related to a course.
- Professor preferences. Some professors are only available on certain days of the week and at specific hours. Also, there are some classrooms that professors prefer to not teach in.
- Departmental ownership of rooms. The problem formulation does not address assigning courses to rooms owned by the associated departments.

C. OUTLINE OF THIS WORK

The research presenting in this paper is intended to provide a basis for a solution that will allow NPS to automate the scheduling system while maintaining the flexibility desired by the faculty and students. The remainder of this thesis is organized as follows:

- Chapter II discusses background information and related work in the fields of timetabling and probability collective theory.
- Chapter III describes the specific implementation of probability collective theory to solve the university course timetabling problem.
- Chapter IV displays the results of the implementation.
- Chapter V provides an overall summary and potential future work.

II. BACKGROUND AND RELATED WORK

A. INTRODUCTION

Timetabling is generally set up as an optimization problem. In this chapter, general approaches to optimization, existing timetabling techniques, and probability collective theory are introduced. An introduction to several optimization algorithms can be found in [8].

Optimization is a term describing methods used for minimizing or maximizing an objective function. The objective function is a function which describes the quality of a system for a given set of parameters, and the goal is to find the state of the system (usually a set of parameters) which give the "best" quality, i.e., which minimize or maximize this function. One common approach is called hill-climbing. Hill-climbing picks a starting state and calculates the value of the objective function for this state. Then the algorithm methodologically changes the state in an attempt to increase the objective function. Once a state change no longer increases the objective function, the algorithm halts. The problems with this technique are that it is possible to stop on local maxima and never find the global optimal solution. Several methods such as using random restarts can help overcome this obstacle.

Another improvement over hill-climbing is simulated annealing. Simulate annealing uses hill climbing with a random walk that allows the algorithm to jump to different states. The amount of randomness in the jumps is controlled by a variable called temperature. Temperature begins at a high value allowing for larger distance jumps and ensuring a high degree of exploration early on. As the algorithm progresses, temperature is slowly decreased allowing for more time to be spent analyzing better choices.

Other common optimization methodologies rely on genetic algorithms. Genetic algorithms draw on the theory of evolution by combining "parent states" and mutating to create new single states in an attempt to improve the objective function. The algorithm begins by generating a set of states called the population that are represented by a string

over a finite alphabet. Offspring strings are generated by combining two parent strings at a randomly chosen crossover point. Additionally, a random mutation of the string can also occur. The offspring is then evaluated and if an improvement is achieved, the state is saved. Similar to simulated annealing, genetic algorithms allow for a large amount of exploration early on since the initial population is so diverse. It also allows for optimization in problems without smooth objective functions.

Another variant of hill-climbing is tabu search. Tabu search is based on short term memory. A list of previously visited states that cannot be visited again is maintained allowing this algorithm to escape from local maxima. This approach is effective for some domains but problems can exist when potentially good solutions are added to the tabu list and never explored. By setting criteria that exclude solutions with certain criteria from the list, this problem can be minimized.

Linear programming is a completely different approach to optimization than the previous methods. Linear programming is used when optimizing subject to constraints [9]. The objective function and constraints are defined as linear functions and the combination of these functions defines a feasible region that encompasses all possible solutions to the objective function. Several algorithms such as the Simplex method are then used to determine the best solution in the feasible region. Integer linear programming is a variation of linear programming that forces all variables to be integers.

B. TIMETABLING

Timetabling can be defined as the assignment of events to a limited number of time periods and locations subject to constraints such that goal objectives are achieved to the maximum extent possible. Several approaches to solving timetabling problems have been researched over the last couple of decades. General solutions have been developed as well as solutions that address specific areas such as employee timetabling, examination timetabling, school timetabling, and sports timetabling. In this section, several algorithms dealing with university timetabling are explored.

A good overview of solutions involving annealing techniques is provided in [10]. The authors note that one of the major drawbacks of applying simulated annealing to the

timetabling problem is that it can take an unacceptably long time to develop a good solution. The authors propose a deterministic approximation to simulated annealing called mean-field annealing that attempts to achieve the same quality solution as simulated annealing in a much shorter amount of time. A rule based preprocessor is used to provide a good starting point. The paper also describes three cooling techniques: geometric cooling, adaptive cooling, and adaptive cooling with reheating as a function of cost. These cooling methods attempt to provide a good balance between early exploration and fine-tuning of the timetable later.

The authors of [11] used a three phase approach that was very successful in the 2003 International Timetabling Competition. Phase 1 consisted of first constructing a feasible timetable using graph coloring and maximum matching algorithms. In this phase, events were placed in timeslots ensuring only that no hard constraints were violated. In phase 2, the algorithm attempts to sequence the events in a way that minimized the soft constraints. A solution space is created that includes all possible permutations of the timeslots and then swaps two entries of the permutation at a time to determine the best combination. Phase 3 uses simulated annealing to optimize the schedule. This algorithm has been very successful and after some modification was shown to be superior to the winner of the 2003 competition.

Chirandini et al. describe a hybrid algorithm in [12] and [13] that was also very successful. The authors tested about 1185 configurations of different combinations of algorithms to determine the best approach. Their best combination included the use of local search, tabu search, and simulated annealing. In the first phase, all hard constraints are attempted to be satisfied by using a local search algorithm. If this search results in local optima, a tabu search is used to exit the local optima and continue with the local search. Next, local search and simulated annealing are used to minimize the number of soft constraints. The algorithm was tested against the problem instances of the 2003 International Timetabling Competition and scored better than the contest winner for all 20 problem instances.

A tabu search hyperheuristic approach is outlined in [14]. The hyperheuristic algorithm takes several low-level heuristics as inputs and based on the evaluation

function determines the best heuristic to use for the given optimization problem. The method involves running each heuristic and assigning a rank based on the effectiveness of that heuristic. The heuristics are then added to a tabu list in order of rank and the optimization is run again using heuristics in order of rank from best performing to worst performing. Ranks are recalculated and the order of the tabu list is adjusted. The process continues until a satisfactory solution is found. The algorithm has proven to be successful in university course timetabling as well as other problems such as scheduling hospital shifts for nurses.

The authors of [15] detail the use of a memetic algorithm. A memetic uses local search to reduce the space of possible solutions to a subspace of local optima. Genetic algorithms and hill-climbing searches are then used to find the optimal solution from the local optima. The algorithm was tested on actual scheduling data from Nottingham University and was found to be promising but somewhat time intensive.

Integer linear programming was used in the approaches described in [16]. This optimization approach involves partitioning the classes into subsets of classes called blocks. These blocks are then scheduled in parallel. The paper deals with the creation of these partitions and also the assignment of students to the blocks. Integer linear programming is used for the optimization and results from tests done on real world high school data show that solutions can be developed in a very short period of time.

A basic agent based approach is provided in [17]. Agents are assigned to each hard and soft constraint in the problem set. In the first phase, each agent develops a feasible schedule based only on its assigned constraint. The solutions are sent to the other agents who then evaluate each solution based on its own constraint and a penalty is assigned according to the number of constraint violations. Next each agent creates a small change to the schedule ensuring that feasibility is not violated. These new schedules are again distributed and scored. Feedback is provided to the agents based on whether the change provided a positive of negative change on the penalty score. The generating agent then decides whether to keep or discard the change. The process is repeated until all constraint violations are eliminated and a final course timetable is developed or until the violations cannot be reduced any further.

Each of the approaches detailed above attempt to solve the university course timetabling problem in a different way with varying degrees of success. The problem remains very difficult and to date no one has found an optimal solution that is general enough to be applied to any timetabling problem without significant modification. The majority of these approaches also do not include the additional constraints placed by considering student demand for courses.

C. PROBABILITY COLLECTIVE THEORY

Probability Collective (PC) theory is a relatively new approach to solving optimizations problems. It has been successfully used in areas including flight control [18], airline flight scheduling [19], and internet traffic routing [20]. PC theory is described thoroughly in [21], [22], [23], [24], [25], [26].

PC theory allows for a distribution of the optimization among agents that represent variables in the system. Collective Intelligence (COIN) is utilized to develop a collective of these agents. Each agent selects actions from a predefined set and evaluates the utility of this choice both for itself and for the collective as a whole. The agent makes subsequent choices based on the determined utility until the system reaches an equilibrium state where no improvements can be made by altering agent actions. PC theory draws on ideas from genetic algorithms, simulated annealing, and statistical physics.

In PC theory, each agent has a probability distribution across the actions available to it and these probabilities are updated based on the utility calculation. This extends the traditional COIN approach in the method in which it updates these probabilities. PC theory assumes that each agent is bounded rational and independent and will make choices based only on its own probability distribution. Bounded rational agents balance their choice of best move with a need to explore the system. Independent agents make their decisions without considering the moves of other agents in the system.

One benefit of using PC theory is that since each agent chooses actions independently, the problem can easily be parallelized. Due to the competition

requirements that the program run on only one processor, the full benefits of this parallelization have not yet been realized in our work.

In the next two sections, the optimization approach and solution algorithm presented in [22] are summarized.

1. Optimization Approach

Assuming that each agent is bounded rational and operates in an environment with world utility G, the system equilibrium will be the optimizer of G subject to any constraints imposed. This equilibrium can be found by minimizing the Lagrangian for each agent as a function of the probability distribution associated with the agents' possible actions. The Lagrangian $\mathcal{L}_i\left(q_i\right)$ is given by

$$\mathcal{L}_i\left(q_i\right) = \mathrm{E}\left[G(x_i, x_{(i)})\right] - TS\left(q_i\right)$$

where G is the system objective which depends on the agent i's action x_i and the actions of all other agents, $x_{(i)}$. The probability distribution of agent i is represented by q_i. S is the entropy of this distribution and is given by:

$$S\left(q_i\right) = -\sum_{x_j} q_i\left(x_j\right) \ln q_i\left(x_j\right)$$

T is the temperature of the system and determines the amount of exploration the agent engages in. Each agent attempts to minimize the Lagrangian function $\mathcal{L}_i\left(q_i\right)$, subject to

$$\sum_{x_i} q_i\left(x_i\right) = 1, q_i\left(x_i\right) \geq 0, \forall x_i$$

This ensures that the sum of the probabilities in the probability distribution sum to 1 and that there are no negative probabilities.

When temperature T is high, much weight is given to the entropy component of the equation, which minimizes the Lagrangian by encouraging a uniform distribution and therefore encourages more exploration of the space by the agent. As the temperature

decreases, exploration becomes less important and the agent begins to exploit action choices which are “better” (lower cost/higher utility) than others.

After each iteration, the probabilities are update using Newton updating, with the update equation

$$q_i(x_i) \to q_i(x_i) - \alpha q_i(x_i) \times \left\{ \frac{\mathrm{E}[G \mid x_i] - \mathrm{E}[G]}{T} + S(q_i) + \ln q_i(x_i) \right\} \tag{1}$$

where α is a step size determining how much the existing probability is modified by this iteration’s results. The probability distribution is then renormalized ensuring that there are no negative probabilities and that the sum all of the probabilities is 1.

To calculate the expected utility for each agent, we use

$$\mathrm{E}(g_i \mid x_i = j) = \frac{N_{ij}^{(k)}}{D_{ij}^{(k)}} = \frac{\sum_m g_i(x_i = j, x_{(i)}) 1(x_i = j) + \gamma N_{ij}^{(k-1)}}{\sum_m 1(x_i = j) + \gamma D_{ij}^{(k-1)}} \tag{2}$$

where $1(x_i = j)$ equals 1 when $x_i = j$ and 0 otherwise. The agent’s private utility is represented by g_i. D tracks the number of times an agent i chooses a particular choice j and N tracks the private utility when then choice is made. Data aging is controlled by γ.

Although not yet used in our system, constraints can be added to the system by the addition of Lagrange multipliers, λ_j, to the global utility along with constraint functions, $c_j(\vec{x})$, as

$$G(\vec{x}) \to G(\vec{x}) + \sum_j \lambda_j c_j(\vec{x}).$$

The update rule for the Lagrange multipliers is

$$\lambda_j \to \lambda_j + \eta \mathrm{E}\left[c_j(\vec{x})\right] \tag{3}$$

where η is separate step size.

Expected utilities for each agent are computed by Monte-Carlo simulation. This is accomplished by all agents repeatedly identically and independently sampling their distributions to generate moves, and then calculating utilities based on these moves. The private utility calculation should be chosen to ensure low bias and low variance. Low bias ensures that the private utility closely resembles the global utility. Low variance ensures that each agent's contribution to global utility is distinguishable.

2. Solution Algorithm

The basic algorithm to solve problems with PC theory is as follows:

a. Initialize the system

 a. Initialize the parameters {T, α, γ}. Set the convergence criteria δ.

 b. Select the number of Monte Carlo Samples.

 c. Initialize the probability collectives

b. While $\delta \geq \left\|\vec{\lambda}^{k} - \vec{\lambda}^{k-1}\right\| + \sum_{i}\left\|\vec{q}_{i}^{\,k} - \vec{q}_{i}^{\,k-1}\right\|$

 1. For each Monte-Carlo sample,

  i. Jointly IID the sample

  ii. Evaluate the objective function

  iii. Compute each agent's private utility

 2. Compute the expected utility for each agent using Eq. (2).

 3. Update the probability distributions using Eq. (1).

 4. Update the Lagrangian multipliers using Eq. (3).

c. Final Evaluation

 1. Determine the highest probability value for each variable

 2. Evaluate the objective function with this set of values.

D. CONCLUSION

In this chapter, the topics of optimization, timetabling, and probability collective theory were discussed. This material formed the basis for that approach used in this thesis. The timetabling section defined and constrained the optimization problem while the probability collective section provided the background for the approach used.

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY

In this chapter we discuss the methodology used in our experiments. In particular, we applied PC theory to attempt to solve the university course timetabling problem. Several modifications to the general probability collective algorithm were made to map the theory to this specific problem.

A. APPLICATION OF PC THEORY TO TIMETABLING

In this section, the details of how the post enrollment course timetabling problem was approached using PC theory are described. The basic algorithm was modified slightly for the purposes of this implementation due to the relative size of the given problem set compared to prior problems implemented with PC theory. In this application, each event is represented by an agent. The available moves are determined individually based on the specific requirements of each event.

In our system, an agent's private utility is calculated by counting the number of collisions that exist given a certain choice. Collisions can occur in two ways. First, if an event is scheduled in the same timeslot and room as another event, the number collisions are equal to the size of the union of the two sets of students. The second form of a collision is when two events are scheduled in the same timeslot but in different rooms and the intersection of the two sets of students is non-zero. The size of the intersection is the number of collisions in this case.

Due to the size of the problem set, global utility is never actually calculated. Instead, an average of the localized global utility is used to estimate actual global utility. Localized global utility is calculated by summing the products of the probability of an event choosing a certain timeslot-room combination and the expected utility associated with that choice.

1. Initialization

During the system initialization data structures are generated representing the events, rooms, and students using data provided by the International Timetabling Competition. We map students to events, rooms to available features, required features to events, events to available time slots, and evaluate event precedence. Initial parameters, including the rate of cooling (or rate of change of temperature) (ΔT), the number of Monte-Carlo samples (m) per iteration, and initial temperature, are set according to user input. Initial temperature is carefully selected to ensure a good amount of exploration occurs early on. The rate of cooling controls how rapidly the system moves from exploration to trade off exploitation.

Next, each event is initialized. The probability distribution is set to only include rooms of the appropriate size, rooms that have the features required by that event, and timeslots that are available to that event. The distribution is initialized by assigning an equal probability to each available choice. Data structures to track utilities for individual choices and the number of times each choice is made (N and D respectively) are created and initialized to zero.

Instead of setting the step size α and data aging factor γ to a preset value, each event sets its own values based on the number of students assigned to that event. Smaller classes are assigned a lower value for α and γ to allow for more movement for a given iteration while larger classes are given higher values to prevent large jumps between iterations. The theory behind this is based on how particles react in the real world. For a given energy imparted on a system, smaller particles will move faster and farther than larger ones. The end result is that events with larger class sizes tend to be placed earlier while the smaller events are allowed more freedom to move and find an optimum slot.

The final phase of event initialization consists of pre-calculating the number of collisions that can occur between two distinct events, by calculating the student set intersection between every pair of events.

2. The Optimization

For every iteration of the optimization, the *m* Monte-Carlo samples for each event (representing room and time slot choices for the next *m* iterations) are first generated. We used stochastic low variance sampling [27, p. 108] to generate all *m* samples up front in O(*m*) time. These samples are initially ordered and must be randomly permuted.

Next, we iterate *m* times. For each iteration, each agent computes its private utility (number of student/course collisions) for its chosen room and time slot, and tracks the number of times each particular choice is made. After all iterations, each event updates its *N* and *D* data structures and calculates expected utility using Eq. [2]. They also calculate their localized global utility and entropy.

If the change in average localized global utility $\geq \delta$, temperature is decreased by ΔT and next set of Monte-Carlo samples is calculated. If the change in average localized global utility is minimal, then the algorithm moves to the final evaluation phase.

3. Final Evaluation

At this point each event should have a probability distribution that reflects the best choice or choices of timeslot-room combinations that allow it to minimize the total number of system collisions. The next step is to assign events to the timeslot-room combination that best suits the event while ensuring that none of the hard constraints are violated. Since it is likely that in a very dense schedule there may still be constraint violations, events are scheduled in descending order of number of students associated with that event.

For each event, the algorithm attempts to schedule the event in the timeslot-room combination with the highest probability. This choice must be compared with already scheduled events to ensure that there are no student collisions, no two events are scheduled in the same room at the same time, and that all events are scheduled in the order established by any precedence requirements. Lower probability choices are tried if the first choice is not successful. If no acceptable timeslot-room combination is found, the event is unscheduled.

In this chapter, the application of probability collective theory to solving the university course timetabling problem was discussed. The modifications to the generic algorithm were outlined.

IV. RESULTS

In this chapter, the results of running the algorithm described in chapter III to solve the university course timetabling problem are described. Additionally, an analysis of the algorithm including sensitivity analysis of the individual variables is provided.

A. RESULTS FOR TIMETABLING COMPETITION INSTANCE FILES

Sixteen problem instance data sets were provided by the International Timetabling Competition for testing purposes. Table 1 lists the number of events, rooms, possible room features, and number of students associated with each instance.

Instance	Events	Rooms	Features	Students
1	400	10	10	500
2	400	10	10	500
3	200	20	10	1000
4	200	20	10	1000
5	400	20	20	300
6	400	20	20	300
7	200	20	20	500
8	200	20	20	500
9	400	10	20	500
10	400	10	20	500
11	200	10	10	1000
12	200	10	10	1000
13	400	20	10	300
14	400	20	10	300
15	200	10	20	500
16	200	10	20	500

Table 1. Description of problem instances.

For the competition, a time limit was imposed based on the number of computer cycles. A benchmark tool was provided on the competition website. The test machine was a virtual machine running Windows XP and allocated 1024 MB of RAM. The base machine was a Macintosh iMac running at 2.8GHz. Based on the benchmark tool, the maximum allowable run time was approximately 600 seconds.

The following parameters were used for the competition: ΔT = 0.90, 500 Monte-Carlo samples per iteration, and an initial temperature factor of 2.0. The determination of α and γ was made based on the actual minimum event student size of 1 and a maximum of 98. The formula to calculate both variables ensured that the values for the largest event were fixed at 0.9 and the variables for the smallest event were fixed at 0.1. The resulting formula is:

$$\alpha = \gamma = 0.00825x + 0.0964$$

where x is the number of students requesting the event.

Table 2 displays the results of the competition runs. The worst case possible column indicates the Distance to Feasibility if no events were placed. A baseline set of data was generated by running the scheduling program with all of the probabilities in the events' collectives distributed uniformly. The program generated valid timetables but was unable to find a feasible solution for any of the problem instances. Each instance is known to have at least one feasible solution though the competition organizers feel that these solutions will most likely not be found in the give time. At the time of writing this thesis, the results from the competition were not available and therefore a comparison with the other competition entries is not yet possible.

Instance	Baseline Distance to Feasibility	Distance to Feasibility	Soft Cost	Run Time (sec)	Worst Case Possible	Baseline Percent Scheduled	Percent Scheduled
1	4735	3076	1798	597	10515	55.0	70.7
2	4982	3170	1725	586	10515	52.6	70.0
3	4483	1997	3010	145	13383	66.5	85.1
4	4987	2040	2711	145	13396	62.8	84.8
5	2525	1239	1157	539	6275	59.8	80.3
6	1814	971	1322	529	6218	70.8	84.4
7	1965	687	1449	133	6733	70.8	89.8
8	1999	756	1496	138	6916	71.1	89.1
9	4962	2814	2076	593	10714	53.7	73.7
10	5235	3035	1685	581	10492	50.1	71.1
11	4116	2804	2937	159	13608	69.8	79.4
12	3967	2930	3123	155	13607	70.8	78.5
13	2967	1424	1175	535	6358	53.3	77.6
14	2772	1362	1154	538	6257	55.7	78.2
15	2122	808	1362	142	6527	67.5	87.1
16	1943	576	1366	145	6819	71.5	91.6

Table 2. Competition results.

B. ANALYSIS OF THE ALGORITHM

Figure 1 shows the progression of average localized global utility over time. Three different problem instances of varying size are shown. In this implementation, utility is actually a cost and lower average costs are desired. The initial high temperature allows for greater exploration in the beginning of the optimization which explains the initial rise in average localized global utility followed by a rapid drop to nearly zero as the events find their optimal timeslot-room combination.

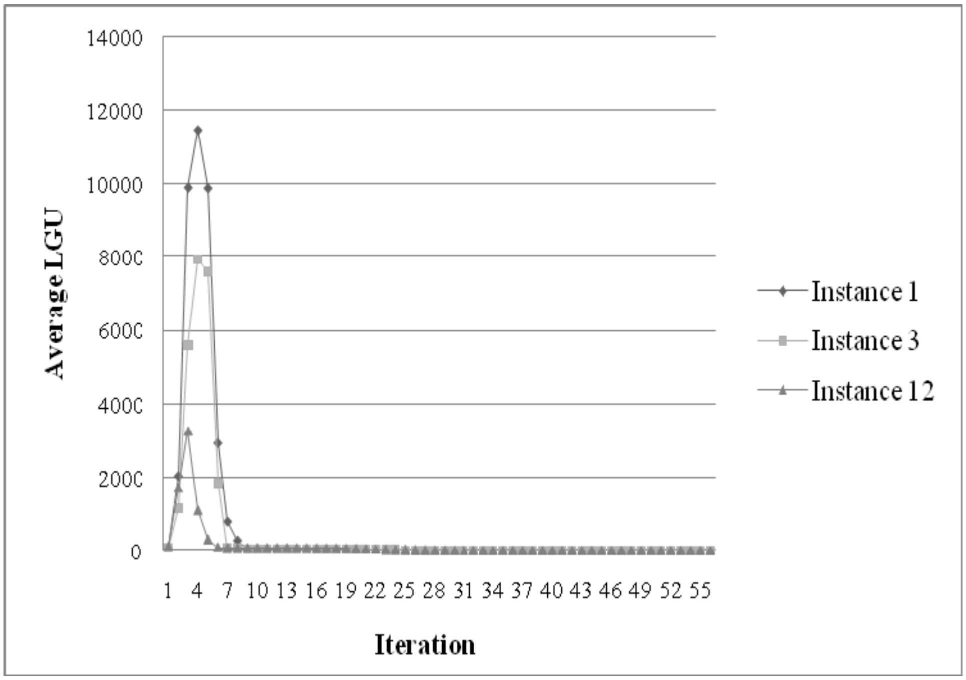

Figure 1. Change in average localized global utility by iteration.

Figures 2 and 3 show the progression of the probability collectives through the optimization process for events 5 and 21 of problem instance 3. Each line represents the change in the probability for one timeslot-room combination over time. Student sizes for the events were 65 and 15 respectively. By iteration 37, event 5 narrowed the number of possible timeslot room combinations to one. At the end of the optimization, event 21 had 3 timeslot-room combination choices of about equal quality.

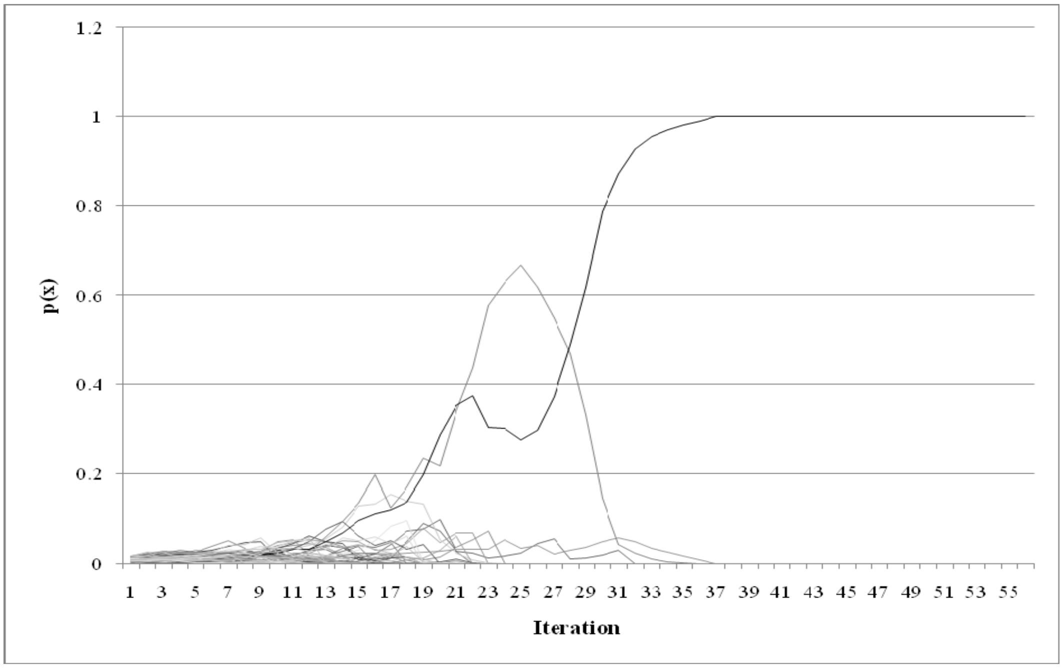

Figure 2. Evolution of the probability collective for event 5 of problem instance 3.

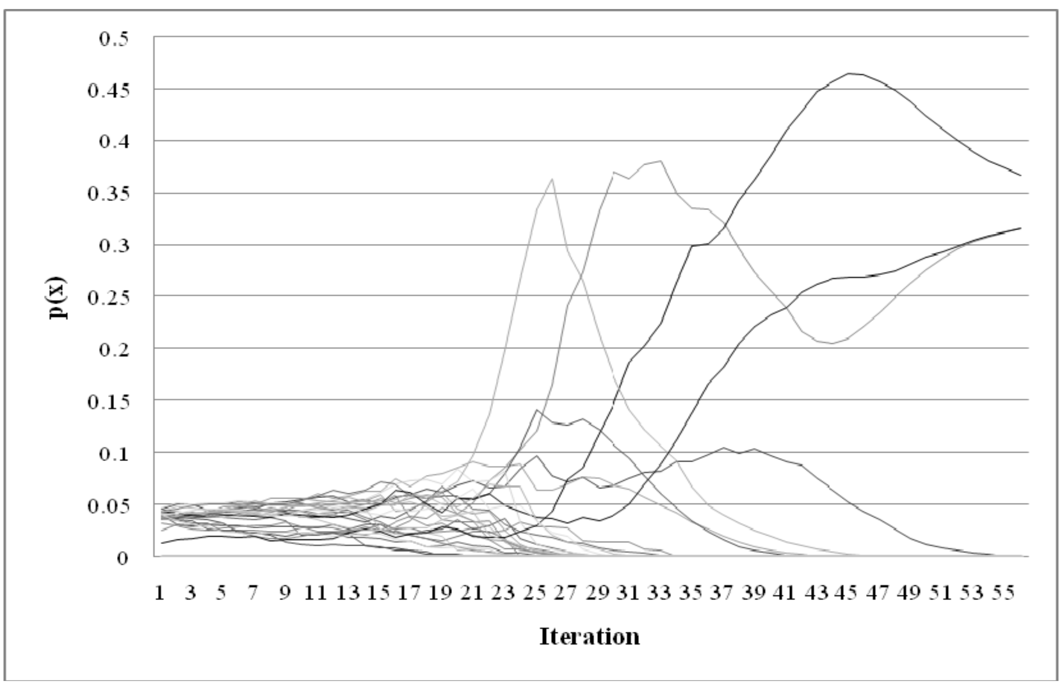

Figure 3. Evolution of the probability collective for event 21 of problem instance 3.

1. Sensitivity Analysis

A sensitivity analysis was conducted on several of the algorithm variables to determine the effect of changing a single variable. The variables chosen were α, γ, ΔT, and the number of Monte-Carlo samples. The baselines were the same as the competition data runs with $\Delta T = 0.90$ and the number of Monte-Carlo samples = 500. For this analysis, α and γ were fixed to values of 0.5 each instead of using the calculation above to set the values based on class size. All calculations were conducted using the data from problem instance 3.

The number of Monte-Carlo samples (m) determines how many times each event samples its distribution per iteration. A higher number of samples ensure that the system is thoroughly sampled but the trade-off is that this higher number significantly increases run time. Values of 250, 500, 1000, and 2000 were chosen for the analysis. With the number of samples below 250, the algorithm was unable to lower average localized global utility. Figure 4 shows the change in average localized global utility for the various values of m. For all values of m other than 250, the curve is very smooth and reacts as predicted. At m=250, average localized global utility is very unstable through the entire run. For both m=250 and m=500, the peak of the curve is above the graph with the peak for m=250 nearly 10,000,000 and the peak for m=500 nearly 9,000. After iteration 15, the results for m=500, 1000, 2000, and 4000 are nearly identical showing that the additional time for the calculation does not provide a significant return on investment. Figure 5 shows the amount of time required for each value of m. The overlaid trend line indicates that and increase in m causes a linear increase in the run time of the program.

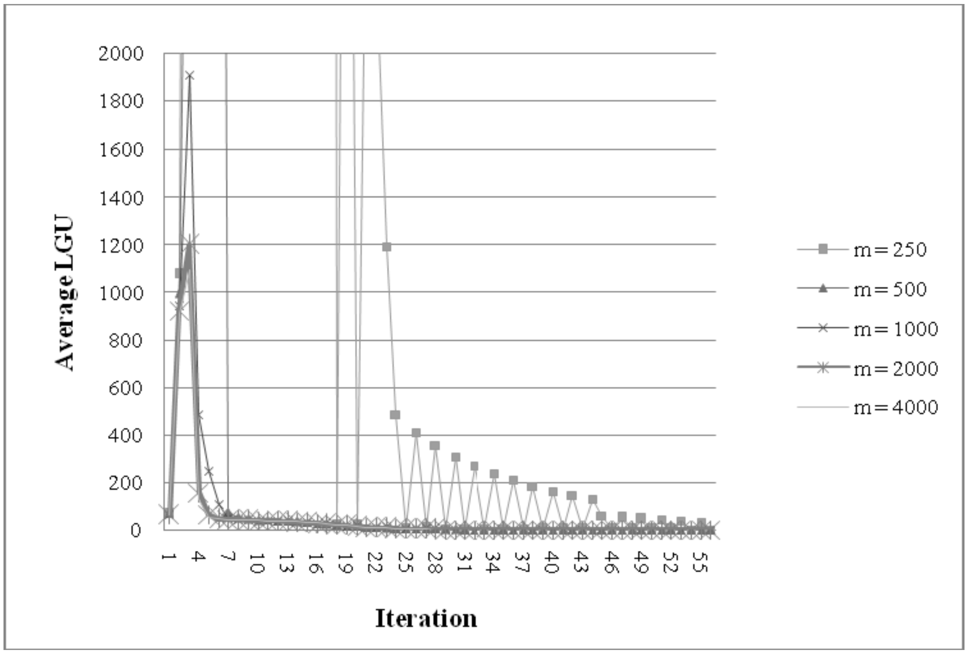

Figure 4. Change in Average Localized Global Utility over time with different values of number of Monte-Carlo samples(m).

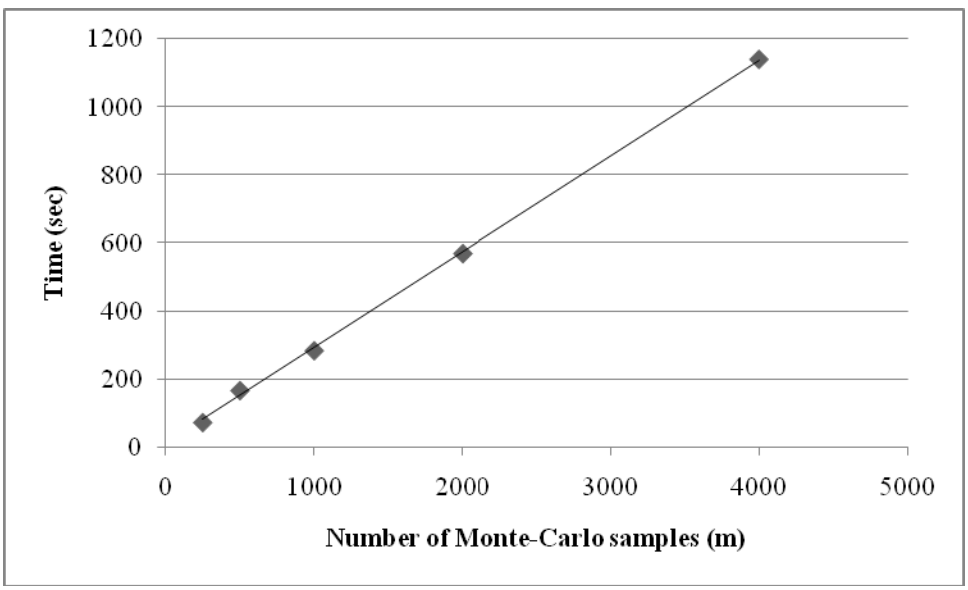

Figure 5. Comparison of number of Monte-Carlo samples to time required

The value of ΔT determines the cooling down length for temperature. A higher value allows less time for exploration and more time for analyzing "good" choices. The downside to reducing temperature to rapidly is that less exploration can lead the

algorithm to get stuck optimizing inside a local minimum and never finding the best solution. Figure 6 shows the results on average localized global utility for the various values of ΔT. For $\Delta T = 0.1$ and 0.5, the program actually got stuck in local minima and was not able to lower average localized global utility below values of 3 and 2, respectively. The resulting timetables had 10 fewer events than the competition result for the same file (163 versus 173 out of 200 possible). Values of $\Delta T = 0.8$ and 0.9 appear to allow sufficient exploration while still providing time to evaluate the "good" decisions. For $\Delta T = 0.95$, average localized global utility dropped at a much slower rate though in the end a greater number of courses were scheduled than the competition result (183 in this case).

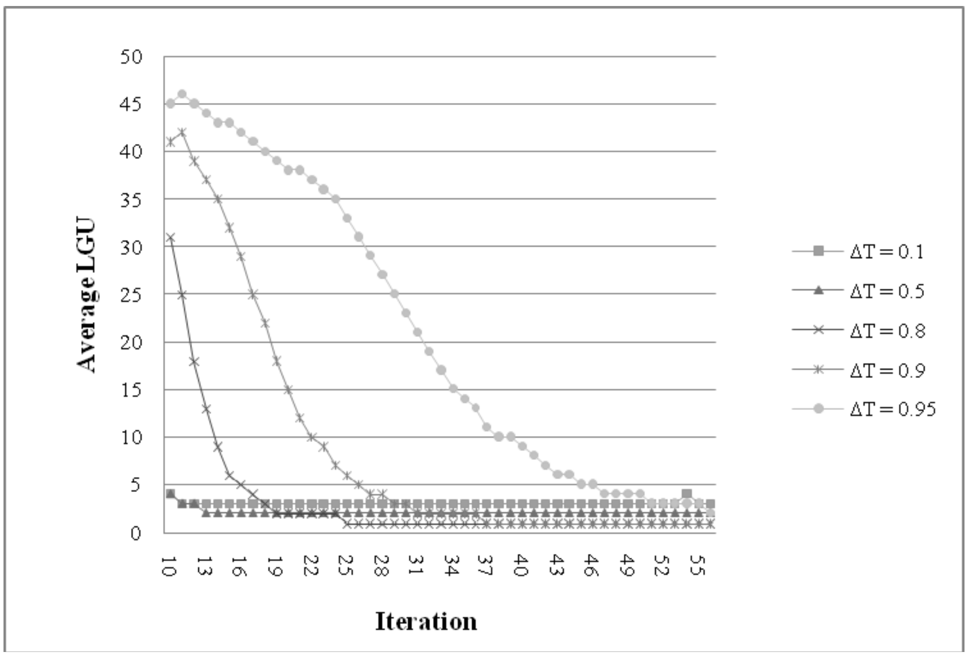

Figure 6. Change in Average Localized Global Utility over time with different values of ΔT.

The magnitude of the previous iteration's values stored in the probability collectives included in the current iteration's calculation of probability is controlled by α. Values of α can range from 0.0 to 1.0 with lower values allowing for less "memory" of old values to be retained. A value of zero means that the new probability collective values are entirely determined by the new data while a value of one gives the past data an equal

weighting with new data. For the analysis, values of 0.1, 0.25, 0.5, 0.75, and 0.9 were chosen. Figure show the change in average localized global utility over time for each of the values of α. Average localized global utility lowered faster with higher values of α, although in each case the values were extremely close by the end.

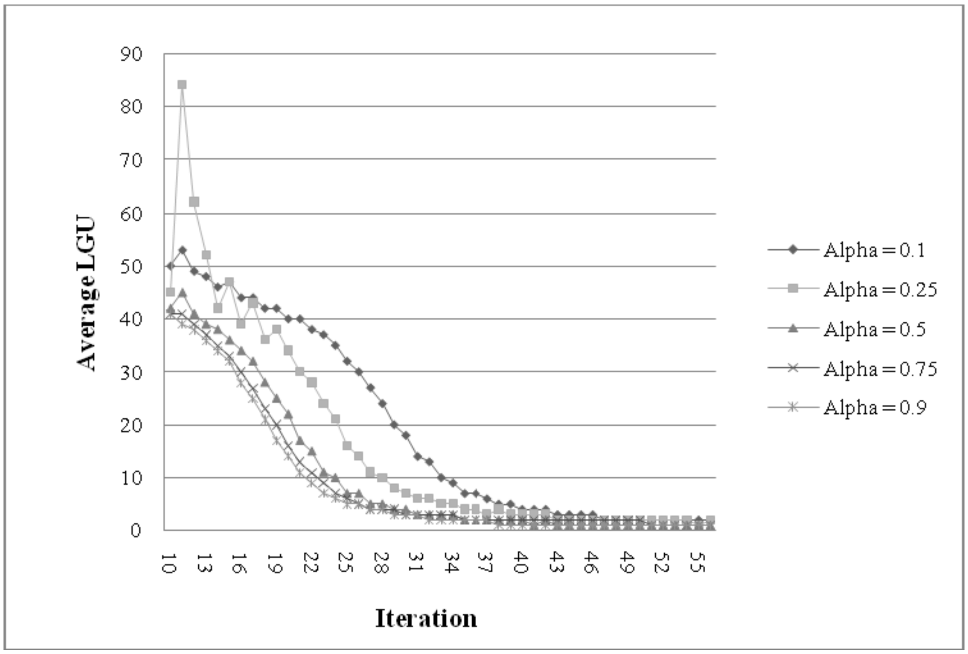

Figure 7. Change in Average Localized Global Utility over time with different values of α.

The data aging factor for the number of collisions stored in each event is controlled by γ. Similar to α, γ can vary from 0.0 to 1.0. A value of zero means that the stored number of collisions is based entirely on the current iteration while a value of one means that the old data has equal weight to the new data. For the analysis, values of 0.1, 0.25, 0.5, 0.75, and 0.9 were chosen. Figure 7 shows that for all values of gamma, the drop in average localized global utility is very similar, though lower values of γ did result in slightly faster drops.

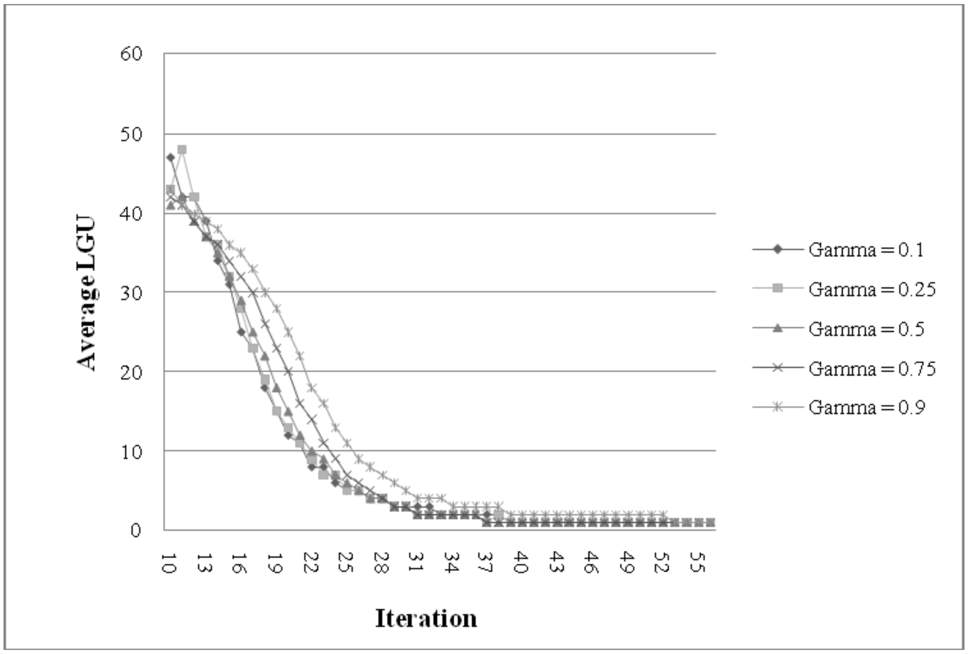

Figure 8. Change in Average Localized Global Utility over time with different values of γ.

C. FINAL DATA RUN

For the competition, a limit was imposed on the amount of time the program was allowed to run (approximately 10 minutes on the test platform used). This limitation was for judging purposes only and in a real world scheduling scenario much more time would be allocated to the optimization. Additionally, the random seed was required to be fixed for reproducibility. An additional data runs was made in an attempt to improve upon the competition results.

For the final run, all parameters were the same as in the competition except that the number of Monte-Carlo samples was increased to 1000 and ΔT was increased to 0.95. Table 3 summarizes the results of this data run. The percent change column compares the Distance to Feasibility change between this run and the completion run. Although the majority of the runs showed improvements, a few runs were actually worse. The best improvement was problem instance 13 showing a gain of 134.6% of scheduled student event requests. Problem instance 7 showed a significant decrease in schedule quality. The

expectation was that by increased the number of Monte-Carlo samples and slowing down the cooling rate that the results would improve. For the majority of the instance problems this improvement was observed. For the other instance problems, it is possible that the algorithm is still getting stuck in local optima or possibly just that the instance is so densely populated that the algorithm is not effective.

Instance	Distance to Feasibility	Soft Cost	Run Time (sec)	Worst Case Possible	Previous Percent Scheduled	Final Percent Scheduled	Percent Change
1	2823	1810	2656	10515	70.7	73.2	109.0
2	3223	1617	2561	10515	70.0	69.3	98.4
3	1606	2608	637	13383	85.1	88.0	124.3
4	1906	2925	631	13396	84.8	85.8	107.0
5	1031	1246	2350	6275	80.3	83.6	120.2
6	769	1313	2332	6218	84.4	87.6	126.3
7	980	1394	606	6733	89.8	85.4	70.1
8	765	1499	621	6916	89.1	88.9	98.8
9	2525	2070	2620	10714	73.7	76.4	111.4
10	2990	1759	2602	10492	71.1	71.5	101.5
11	2657	3018	701	13608	79.4	80.5	105.5
12	3084	3113	709	13607	78.5	77.3	95.0
13	1058	1180	2344	6358	77.6	83.4	134.6
14	1113	1197	2343	6257	78.2	82.2	122.4
15	788	1406	659	6527	87.1	87.9	102.5
16	523	1496	666	6819	91.6	92.3	110.1

Table 3. Data run results with m = 1000 and $\Delta T = 0.95$

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

We have implemented a solution to the post enrollment course timetabling problem based on Probability Collectives theory. The algorithm produced valid timetables for every instance, though 100% student placement was not achieved for any instance. The algorithm was able to successfully schedule between 70.0% and 91.6% of the student event requests in the time allowed by the competition rules. Though the algorithm was not as effective as desired, compared to the baseline data the results were considerably better. The algorithm scheduled between 9.6% and 24.3% more student event requests than the algorithm with uniformly distributed probability distributions. Additional gains were made by relaxing the time requirements, increasing the number of Monte-Carlo samples, and lowering the cool down rate.

B. FUTURE WORK

The research presented in this thesis provides a basis for solving timetabling optimization problems. Several modifications need to be made prior to being able to apply the algorithm to the specific NPS timetabling problem. Several additional constraints must be added to allow the algorithm to deal with all of the requirements that the current schedulers must handle. Examples of these constraints include multiple sections for courses and courses that also have labs associated with them. Additional hard and soft constraints may better push the optimization in a particular direction, so a study of the effect of these additional constraints on the optimization should be conducted.

Alternate cooling schemes might improve the search optimization. The current scheme incrementally reduces temperature from a base value. A more sophisticated method of cooling might help escape local minima more efficiently providing the events a more accurate probability distribution.

The current algorithm uses a localized global utility calculated by each agent to estimate the individual agent's contribution to the overall quality of the timetable. This method was chosen because of the significant time required to calculate the true global utility. A more accurate calculation of world utility may improve the effectiveness of the algorithm. Additionally, a fairly naive method was used to calculate private utility. The current method simply counts the number of "collisions" between events scheduled in the same timeslot. A collision occurs when a student is scheduled for two events in the same timeslot. Two examples of slightly more sophisticated methods to calculate private utility, Team Game (TG) and Wonderful Life Utility (WLU), are discussed in [20]. Team Game sets the local utility equal to the global utility while WLU clamps each agent's action, normally to the one with the lowest probability, and calculates a modified global utility based on these actions. Both calculations result in utilities with low bias and variance.

One of the major benefits of PC theory is the inherent parallelism. A modification of algorithm so that the individual agents can be distributed across several processors may provide significant improvements in program run times. Also, the current implementation was written in Java. It is possible that implementing the algorithm in a more efficient language could lower run times and more easily facilitate the use of parallel processing.

LIST OF REFERENCES

[1] F. J. Hederra, "Timetabling Courses at the Naval Postgraduate School," 1994.

[2] I. Dikmen, D. Stewart and M. Verett, "Business process re-engineering," 2005.

[3] P. Franses and G. Post, "Personnel scheduling in laboratories." in *PATAT; Lecture Notes in Computer Science,* 2002, pp. 113-119.

[4] C. A. White and G. M. White, "Scheduling doctors for clinical training unit rounds using tabu optimization." in *PATAT; Lecture Notes in Computer Science,* 2002, pp. 120-128.

[5] M. A. Trick, "Integer and constraint programming approaches for round-robin tournament scheduling." in *PATAT; Lecture Notes in Computer Science,* 2002, pp. 63-77.

[6] C. Barnhart and S. Shen, "Logistics service network design for time-critical delivery." in *PATAT; Lecture Notes in Computer Science,* 2004, pp. 86-105.

[7] R. Lewis, B. Paechter and B. McCollum, "Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition," Cardiff University, Cardiff Business School, Accounting and Finance Section, Jul 2007.

[8] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* ,2nd international ed. ed.Upper Saddle River: Prentice Hall, 2003.

[9] R. L. Rardin, *Optimization in Operations Research.* Upper Saddle River, NJ: Prentice Hall, 2003.

[10] M. A. S. Elmohamed, P. D. Coddington and G. Fox, "A comparison of annealing techniques for academic course scheduling," in *PATAT '97: Selected Papers from the Second International Conference on Practice and Theory of Automated Timetabling II,* 1998, pp. 92-114.

[11] P. Kostuch, "The University Course Timetabling Problem With a Three-Phase Approach," *The Practice and Theory of Automated Timetabling V (PATAT'04, Selected Papers). Lecture Notes in Computer Science,* vol. 3616, pp. 109-125, 2004.

[12] M. Chiarandini, M. Birattari, K. Socha and O. Rossi-Doria, "An Effective Hybrid Algorithm for University Course Timetabling," *Journal of Scheduling,* vol. 9, pp. 403, 2006.

[13] M. Chiarandini, K. Socha, M. Birattari and R. O. Doria, "International timetabling competition. A hybrid approach," FG Intellektik, FB Informatik, TU Darmstadt, Germany, 2003.

[14] E. K. Burke, G. Kendall and E. Soubeiga, "A Tabu-Search Hyperheuristic for Timetabling and Rostering," *J. Heuristics,* vol. 9, pp. 451-470, 2003.

[15] E. K. Burke, J. P. Newall and R. F. Weare, "A memetic algorithm for university exam timetabling," in *Selected Papers from the First International Conference on Practice and Theory of Automated Timetabling,* 1996, pp. 241-250.

[16] N. Boland, B. D. Hughes, L. T. G. Merlot and P. J. Stuckey, "New integer linear programming approaches for course timetabling," *Comput. Oper. Res.,* vol. 35, pp. 2209-2233, 2008.

[17] Y. Yang, R. Paranjape and L. Benedicenti, "An agent based general solution model for the course timetabling problem," in *AAMAS '06: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems,* 2006, pp. 1430-1432.

[18] S. Bieniawski, "Distributed Optimization and Flight Control Using Collectives," 2005.

[19] N. E. Antoine, S. Bieniawski, D. H. Wolpert and I. Kroo, "Fleet Assignement Using Collective Intelligence," *Proceedings of. 42nd Aerospace Sciences Meeting,* vol. 0622, 2004.

[20] C. Huang, S. Bieniawski, D. H. Wolpert and C. E. M. Strauss, "A comparative study of probability collectives based multi-agent systems and genetic algorithms," in *GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation,* 2005, pp. 751-752.

[21] S. R. Bieniawski and D. H. Wolpert, "Product Distributions for Distributed Optimization," *International Conference on Complex Systems 2004,* 2004.

[22] S. R. Bieniawski, I. M. Kroo and D. H. Wolpert, "Discrete, Continuous, and Constrained Optimization Using Collectives," *10th AIAA/ISSMO Multi-Disciplinary Analysis and Optimization Conference,* 2004.

[23] S. Bieniawski and D. H. Wolpert, "Adaptive, distributed control of constrained multi-agent systems," in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems,* 2004, pp. 1230-1231.

[24] C. F. Lee and D. H. Wolpert, "Product distribution theory for control of multi-agent systems," in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems,* 2004, pp. 522-529.

[25] K. Tumer and D. Wolpert, "A survey of collectives," in *Collectives and the Design of Complex Systems* ,1st ed.K. Tumer and D. Wolpert, Eds. New York: Springer, 2004, p. 1.

[26] K. Tumer and D. Wolpert, "The theory of collectives," in *Collectives and the Design of Complex Systems* ,1st ed.K. Tumer and D. Wolpert, Eds. New York: Springer, 2004, p. 1.

[27] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics.* Cambridge, Massachusetts: The MIT Press, 2005.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California